

Exam

EDAF80 Computer Graphics : Introduction to 3D

2023–10–26, 08.00–13.00

Please answer in English.

Please answer each question on a separate page.

Dictionaries for English are allowed.

Electronic calculators are **not** allowed.

Grading: The maximum score is 6.0. A score of 3.0 or above is needed to pass.

1. Transformations

- a) Given the three matrices **A**: a non-uniform scaling with 3 in x , 5 in y and 9 in z , **B**: rotation 90 degrees around the x -axis and **C**: translation by the vector $\mathbf{v} = (7, 0, 4)$. Given the point $P = (1, 1, 1)$, what is the location of $P' = \mathbf{BAC}P$? (0.3p)
- b) Vertex positions and normal vectors are transformed in the Vertex Shader. When should they be transformed differently, and why? (0.2p)
- c) The GPU can automatically discard triangles that are facing away from the camera. What is this function called, and how is it determined? (0.2p)
- d) Describe the transform **M** needed to move the triangle A to A' and the transform **N** that moves B to B' in Figure 1 below. You can use the syntax used for the Node class in Assignment 1, e.g. `Translate(x, y)`, `RotationZ(theta)`, `Scale(x, y)`. Include in your answer the order of operations. Assume that the origin is placed at the lower left corner of triangle A, and all operations operate in the XY plane. (0.3p)

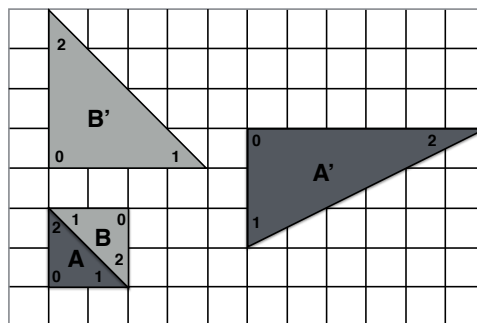


Figure 1: Transform from A to A' and B to B'

2. GLSL

- a) Vectors that are output from the Vertex Shader are used as inputs to the Pixel Shader. What must be done to these vectors before they are used for lighting in the Pixel Shader, and Why? (0.1p)
- b) The OpenGL Shading Language (GLSL) includes the `texture(S,I)` command. Give an example of using the command with a `vec2` as input, and another example that uses a `vec3` as input. Explain what the two parameters are for each usage. (0.2p)
- c) Describe what the GLSL type qualifiers, `uniform` and `const` do, and how they are used in GLSL. What is the difference between them? (0.2p)
- d) Describe which vectors are used as input to the GLSL `refract(...)` function. (0.2p)

- e) Name three shader compiler errors in the following Vertex and Pixel shader pair. (0.3p)

Vertex shader:

```
in vec4 vPosition;
in vec3 vertexColor;
out vec3 color;
void main() {
    color = vertexColor;
}
```

Pixel shader:

```
in vec3 color;
in vec4 fColor;
void main() {
    fColor = vec4(color);
}
```

3. Normal Mapping, Homogeneous coordinates, and Projection

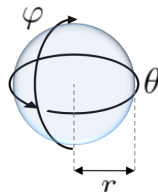


Figure 2: A parametric representation of a sphere.

- a) Given the parametric representation of a sphere shown in Figure 2, with a radius of ($r = 1$), the parametric form is given by

$$s(\theta, \varphi) = \begin{bmatrix} \sin \theta \sin \varphi \\ -\cos \varphi \\ \cos \theta \sin \varphi \end{bmatrix},$$

where $\theta \in [0, 2\pi]$ and $\varphi \in [0, \pi]$. We can construct a tangent space at the point, s , where the tangent, \mathbf{t} , and binormal, \mathbf{b} , are

$$\mathbf{t} = \begin{bmatrix} \cos \theta \sin \varphi \\ 0 \\ -\sin \theta \sin \varphi \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \sin \theta \cos \varphi \\ \sin \varphi \\ \cos \theta \cos \varphi \end{bmatrix}.$$

how do we compute the normal, \mathbf{n} ?

(0.1p)

- b) Given the normal, \mathbf{n} , is

$$\mathbf{n} = \begin{bmatrix} \sin \theta \sin \varphi \\ -\cos \varphi \\ \cos \theta \sin \varphi \end{bmatrix}.$$

Compute the position and tangent space at the parametric value $(\theta, \varphi) = (\frac{3\pi}{2}, \frac{\pi}{2})$, given $\sin(\frac{3\pi}{2}) = -1, \cos(\frac{3\pi}{2}) = 0, \sin(\frac{\pi}{2}) = 1, \cos(\frac{\pi}{2}) = 0$.

(0.2p)

- c) For the point, \mathbf{s} , and normal, \mathbf{n} , derived in the previous question, we perform a texture lookup into a normal map texture. If the texture lookup returns the color represented by (255, 128, 192) in 8-bit, calculate the perturbed normal in object space? (0.3p)
- d) What are homogeneous coordinates and why are they used in the graphics pipeline? (0.2p)
- e) What four variables that describe the shape of the viewing frustum are used to compute the components of the OpenGL perspective projection matrix? (0.2p)

4. Shaders, reflection and OpenGL

- a) Explain what Flat, Gouraud and Phong Shading are by comparing how they are calculated. (0.2)
- b) How do you implement reflection mapping that reflects a background that surrounds a scene using a skybox? Explain the GLSL functions needed in the Pixel Shader, their inputs and outputs, and any other setup. (0.3p)
- c) What direction does the conventional OpenGL camera point in? (0.1p)
- d) Name three algorithmic or compiler errors in the following Vertex shader. (0.3p)

```
in vec4 vPos;
in vec3 vNormal;
uniform float time;
uniform mat3 MVP;
void main()
{
    float scale = 1.0 + sin(time);
    vec3  pnew = vPos.xyz + vNormal.scale;
    gl_Position = MVP*vec4(pnew);
}
```

- e) If the Vertex Shader in the previous question was correct, what effect would it have on a sphere? (0.1p)

5. Graphics Pipeline

- a) Give the three main stages of the Graphics Pipeline. Describe what these stages do, including their input and output values. (0.3p)
- b) For each stage of the Graphics Pipeline, what properties of a 3D scene put a heavy load on each stage. (0.3p)
- c) A vertex is transformed from object space to screen space by the graphics pipeline. A step in that transformation is the perspective divide. Where in the pipeline does it happen, and what does it do? (0.2p)
- d) Mention at least two effects that are hard to reproduce in real-time graphics, but needed for high quality offline rendering. Motivate your answer. (0.2p)

6. Rasterization and Ray Tracing

- a) Compare how rasterization and ray tracing process triangles and determine visibility per pixel. Include in your answer the difference in how pixels and triangles are processed in the two algorithms. (0.4p)
- b) Shadows, reflections and refractions are easier to implement using ray tracing than using rasterization, as used in OpenGL. Explain why, by describing how the three techniques are implemented in both ray tracing and rasterization and comparing their implementations. (0.6p)

The end.
